# Synthetic Data, Real Errors: How (Not) to Publish and Use Synthetic Data

**Boris van Breugel** [1]   **Zhaozhi Qian** [1]   **Mihaela van der Schaar** [1][2]

## Abstract

Generating synthetic data through generative models is gaining interest in the ML community and beyond, promising a future where datasets can be tailored to individual needs. Unfortunately, synthetic data is usually not perfect, resulting in potential errors in downstream tasks. In this work we explore how the generative process affects the downstream ML task. We show that the naive synthetic data approach—using synthetic data as if it is real—leads to downstream models and analyses that do not generalize well to real data. As a first step towards better ML in the synthetic data regime, we introduce Deep Generative Ensemble (DGE)—a framework inspired by Deep Ensembles that aims to implicitly approximate the posterior distribution over the generative process model parameters. DGE improves downstream model training, evaluation, and uncertainty quantification, vastly outperforming the naive approach on average. The largest improvements are achieved for minority classes and low-density regions of the original data, for which the generative uncertainty is largest.

## 1. Introduction

Data is the foundation of most science. Recent advances in deep generative modelling have seen a steep rise in methods that aim to replace real data with synthetic data. The general idea is that synthetic data resembles the real data, while guaranteeing privacy (Ho et al., 2021; Yoon et al., 2020; Jordon et al., 2019; van Breugel et al., 2023), improving fairness (Xu et al., 2018; 2019a; van Breugel et al., 2021), augmenting the dataset size (Antoniou et al., 2017; Dina et al., 2022; Das et al., 2022; Bing et al., 2022), or simulating distributional shifts (Yoon et al., 2018). Often the aim is to be able to use the synthetic data in place of the real data for
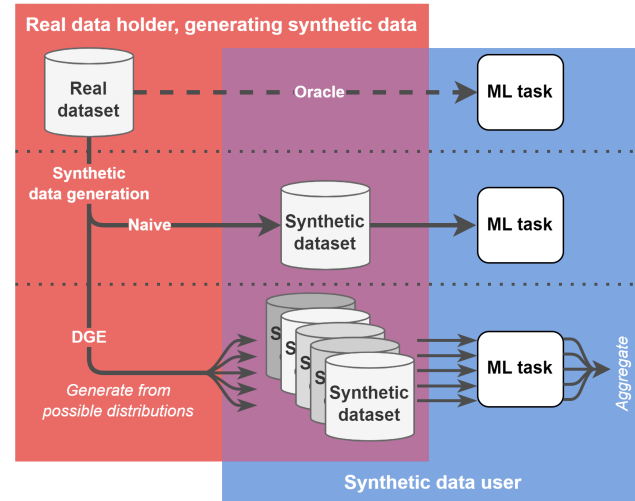


Figure 1. Synthetic data is not perfect, which affects downstream ML tasks, e.g. training a prediction model. The naive synthetic data approach generates one synthetic dataset and treats it like it is real. We propose using an ensemble of generative models for capturing the generative uncertainty, implicitly encoded into different synthetic data distributions.

some downstream task, e.g. statistical analyses or training an ML supervised model. The hope is that downstream results are equally valid in the real-world—e.g. a prediction model trained on synthetic data will do well on real data. Evidently, whether this is true will rely entirely on how well the synthetic data describes the real data.

This brings us to the focus of this work: how do we do good ML on synthetic data, given that the generative process underlying the synthetic data is not perfect. If we are the data publisher, how should we create and publish synthetic data for it to be most useful to downstream users? And if we are the downstream user, can we create models that are more robust to potential errors, evaluate models reliably using synthetic data, and how do we estimate uncertainty for the downstream results? If we envision a future where synthetic data plays a significant role in research, these are pertinent questions.

Let us first highlight why this is an important topic in practice. First, synthetic data is not perfect. Deep generative models may fail in numerable ways, with consequences such

[1]DAMTP, University of Cambridge [2]Alan Turing Institute. Correspondence to: Boris van Breugel <bv292@cam.ac.uk>.

as mode collapse, noisy data, memorisation of training data, and poor coverage (van Breugel & van der Schaar, 2023). Second, in addition to synthetic data being imperfect, even just quantifying the quality of generative models is hard, because this requires comparing distributions as a whole—a notoriously difficult task (Alaa et al., 2022). These two points make it hard to guarantee the data is 'good' or 'bad'. Third, even if we would be able to measure the quality of synthetic data accurately, in general is not at all trivial how we would use this information for estimating the influence of the generative process on the downstream result—e.g. when training a neural network, the influence of individual training samples is highly complex. Since the downstream user usually has no access to real data (see Figure 1), they cannot verify results on real data. Let us give a real example of what could you go wrong when using synthetic data.

*Example.* We take the SEER prostate cancer dataset and generate synthetic data using CTGAN with different numbers of hidden layers. Subsequently, we use a train-test-split on the synthetic data and train a random forest for classification. We compare the train-on-synthetic-test-on-synthetic (TSTS) accuracy and the train-on-synthetic-test-on-real (TSTR) accuracy, the lat-



*Figure 2.* Conclusions drawn from synthetic data do not always transfer to real data.

ter measured on a real hold-out test set (Jordon et al., 2021). Figure 2 displays the results over 10 runs. We see that the real performance of the downstream models is comparable across different generative model depths—an indicator that the utility of the synthetic data is similar. On the other hand, the TSTS performance indicates a large preference for the data generated using the deeper generator, with the corresponding TSTS vastly overestimating the TSTR. Also note the TSTS estimates have an undesirable high variance, due to the added randomness in the generative process.

**Contributions.** Through this work we explore how—and how *not*— to perform basic ML tasks when using synthetic data. The contributions are as follows.

1. We investigate how the standard, naive way of using synthetic data—using it like it is real data—yields poor downstream models, poor downstream model evaluation, and poor uncertainty quantification, due to it ignoring errors in the generation process *itself*.

2. We introduce Deep Generative Ensemble (DGE) as a simple synthetic data framework for alleviating

these concerns through generating multiple synthetic datasets. An advantage of DGE is its flexibility and compatibility with different deep generative base model (e.g. VAE, GAN, diffusion models).

3. We investigate why and when DGE provides better downstream models, evaluation, selection, and better downstream uncertainty quantification.

4. Furthermore, we explore how DGE improves upon the naive approach especially in low-density regions. This is important, since these regions may correspond to underrepresented groups in the population.

Section 3 is mostly targeted at data publishers, focusing on synthetic data issues and how DGE aims to fix these. Section 4 explores experimentally how the naive approach fails, and describes how DGE-generated synthetic data can be used by data users for better downstream ML. In Section 5 we highlight takeaways for both groups.

## 2. Related Work

**Generative ensembles and dropout.** Deep generative models—notoriously GANs (Goodfellow et al., 2014)—often lack diversity in their generated samples, i.e. a GAN that aims to generate patient hospital records may produce a lot of old people but hardly any young people. A large body of work (Tolstikhin et al., 2017; Grover & Ermon, 2017; Ghosh et al., 2017; Hoang et al., 2018) aims to fix diversity issues by ensembling GANs. Some of these approaches use boosting (Tolstikhin et al., 2017; Grover & Ermon, 2017), others use multiple generators (Ghosh et al., 2017; Hoang et al., 2018), discriminators (Nguyen et al., 2017; Ghosh et al., 2017), or dropout (Mordido et al., 2018). These approaches are entirely generative performance focused, and do not consider any application for or insight into improving some downstream ML task. Most importantly, they do not consider how data should be published, and these methods result still in a single synthetic dataset. In Section 4 we explore why publishing a single synthetic dataset does not suffice, even if it is generated by an ensemble.

**Uncertainty quantification in ML.** Uncertainty quantification has gained significant attention in recent deep learning literature, see (Abdar et al., 2021) for an overview. One of the more popular methods is Deep Ensembles (Lakshminarayanan et al., 2016), which provides a straightforward approach to supervised model uncertainty estimation: train multiple networks, create predictions using each network and consider the variance between the different networks. Even though this approach is simple, Deep Ensembles have been shown to perform very positively in comparison to fully bayesian methods, possibly due to their ability to capture uncertainty at a more global level of the weight space
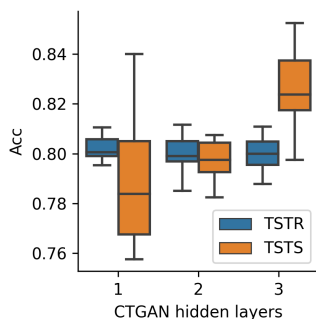
*Table 1.* Comparison to related work. (i) Focuses on generative models, (ii) Considers downstream ML tasks, (iii) Considers error in the generative process, (iv) Provides guidelines to synthetic data publishers and users.

| Method | Works | (i) | (ii) | (iii) | (iv) |
|---|---|---|---|---|---|
| Ensembles of generative models | (Tolstikhin et al., 2017; Grover & Ermon, 2017; Ghosh et al., 2017; Hoang et al., 2018) | ✓ | ✗ | ✗ | ✗ |
| Dropout-GAN | (Mordido et al., 2018) | ✓ | ✗ | ✗ | ✗ |
| Deep Ensembles | (Lakshminarayanan et al., 2016) | ✗ | ✓ | ✗ | ✗ |
| MC dropout | (Gal & Ghahramani, 2015) | ✗ | ✓ | ✗ | ✗ |
| Generative models for UQ | (Böhm et al., 2019; Phan et al., 2019; Sensoy et al., 2020; Liu et al., 2022) | ✓ | ✓ | ✗ | ✗ |
| Deep Generative Ensemble (DGE) | | ✓ | ✓ | ✓ | ✓ |

(Fort et al., 2019). Note that the other very popular UQ method of Monte Carlo dropout (Gal & Ghahramani, 2015) can also be seen as an ensemble method, but where the networks share most of the parameters. To the best of our knowledge, we are the first to apply UQ to generative models and their downstream task. We note that there are works that consider the opposite: applying generative models to UQ (Böhm et al., 2019; Phan et al., 2019; Sensoy et al., 2020; Liu et al., 2022). The aim of these methods is to improve UQ using some form of synthetic data, which is entirely tangential to our work that is interested in the uncertainty in the generative process itself. See Table 1 for an overview.

## 3. Modelling Generative Uncertainty

### 3.1. Set-up

Let $X, Y$ be random variables on $\mathcal{X}, \mathcal{Y}$ denoting features and label, respectively, and let us be given real data $\mathcal{D}_r = (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{n_R}$ from distribution $p_r(X, Y)$. Let $G_\theta$ be a generator parameterised by $\theta$ that outputs samples with distribution $p_\theta(X, Y)$. We denote samples from $G_\theta$ by $\mathcal{D}_s(\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^{n_S}$.

In the typical generative modelling setting, the objective is to minimise:

$$\theta = \arg\min_\theta D(p_\theta, p_r) \qquad (1)$$

for some divergence metric $D$ (e.g. KL-divergence or Wasserstein distance). Though in the limit of infinite training data and capacity some generative models may be guaranteed to achieve $p_\theta = p_r$ (e.g. for GANs (Goodfellow et al., 2014)), in practice $p_\theta$ is only an approximation. Evidently, inaccuracies in $p_\theta$ imply that $\mathcal{D}_s$ has a different distribution as $\mathcal{D}_r$, and hence this affects any downstream task $T$ we may perform using $\mathcal{D}_s$. This task $T$ can depend directly on the synthetic data—e.g. estimating the density at some point—or indirectly—e.g. estimating treatment effects or making prediction by first training some supervised ML model $g$. Thus, the variable $T$ is a random variable

itself, due to it depending on random $\mathcal{D}_s$, as well as possible training randomness (e.g. if it is a prediction of a downstream neural network). In any case, we want to take into account the uncertainty in $\theta$ when performing this task.

### 3.2. Influence of Data on Downstream Task

To account for the synthetic data generation process when computing downstream $T$, let us consider the distribution of $T$. Let us denote the distribution of downstream $T$ w.r.t. the real data $\mathcal{D}_r$ as $p(T|\mathcal{D}_r)$, we can write:

$$p(T|\mathcal{D}_r) = \int p(T|\mathcal{D}_s)p(\mathcal{D}_s|\theta)p(\theta|\mathcal{D}_r)d\mathcal{D}_s d\theta. \qquad (2)$$

Let us look at the right-hand-side terms in turn. $p(T|\mathcal{D}_s)$ is the distribution of $T$ conditional on the synthetic data, which is entirely dependent on the downstream task and something we have no control over as a data publisher. The term $p(\mathcal{D}_s|\theta)$ is the output of the synthetic data generator for some $\theta$, which we can sample from, but usually have no explicit expression for (in case of deep generative models). At last, $p(\theta|\mathcal{D}_r)$ is the distribution over the generative modelling parameters given the real data. This is the term we are most interested in; it reflects the uncertainty we have in the generative model parameters themselves.

Computing the integral in Eq. 2 exactly is intractable for most settings of interest (e.g. if the synthetic data is generated using a GAN). However if a data user would have expressions for all of the terms in Eq. 2, they could use Monte Carlo integration for computing any statistic of interest (e.g. the variance). That is, we sample $\hat{\theta}^k \sim p(\theta|\mathcal{D}_r)$, sample sufficiently large $\mathcal{D}_s^k \sim p(\mathcal{D}_s|\theta^k)$, and sample $\hat{T}^k \sim p(T|\mathcal{D}_s^k)$ for $k = 1, ..., K$, $K \in \mathbb{N}$. This allows us to approximate statistics of interest, for example the empirical mean $\mathrm{E}_{T\sim\hat{p}(T|\mathcal{D}_r)}[T] = \frac{1}{K}\sum_k \hat{T}^k$ and variance $\mathrm{Var}_{T\sim\hat{p}(T|\mathcal{D}_r)}(T) = \frac{1}{K-1}\sum_k(\hat{T}^k - \mathrm{E}_{T\sim\hat{p}(T|\mathcal{D}_r)}[T])^2$ Evidently, there is a trade-off when choosing $K$: a larger $K$ will give more accurate estimates, but also larger computational costs. We will study this further in the experiments.

### 3.3. Modelling the Posterior over $\theta$

So how do we model $p(\theta|\mathcal{D}_r)$? The Bayesian approach could motivate us to parameterise the forward generative process, giving $p(\mathcal{D}_r|\theta) = \prod_i p_\theta(\boldsymbol{z})$, and some prior $p(\theta)$, which would allow computing the posterior over $\theta$: $p(\theta|\mathcal{D}_r) = \frac{p(\theta)p(\mathcal{D}_r|\theta)}{\int p(\theta)p(\mathcal{D}_r|\theta)d\theta}$. Computing the denominator is intractable for deep generative models. Consequently, we need to approximate this further. We draw inspiration from the supervised uncertainty quantification (UQ) literature, which aims to estimate $p(\phi|\mathcal{D})$ for some predictive model parameterised by $\phi$ trained on data $\mathcal{D}$. We borrow a popular technique: Deep Ensembles.

### 3.4. Approximating the Posterior: Deep Generative Ensemble (DGE)

Deep Ensembles (Lakshminarayanan et al., 2016) assumes that we can approximate $p(\theta|\mathcal{D}_r)$ as the empirical distribution over the training process of some deep neural network. In the generative setting, this means we choose a deep generative model class (e.g. VAE, GAN, diffusion model, normalizing flow), train the generative model $K$ times, giving $K$ local solutions $\hat{\theta}^k$ to Eq. 1, and we approximate: $p_{DGE}(\theta|\mathcal{D}_r) = \frac{1}{K}\sum_k \delta(\theta = \hat{\theta}^k)$, after which we can use this distribution for computing any downstream statistic of interest. This is a strong assumption and indeed a crude Bayesian approximation of the true posterior—see (Wilson & Izmailov, 2021) for an in-depth discussion. Nonetheless, Deep Ensembles have a solid track record in predictive UQ, often outperforming more traditional Bayesian methods (Fort et al., 2019).

**Choosing the Baselines.** An advantage of DGE is that it allows for different generative model classes. In this paper we focus on tabular data, because many high-stake applications of synthetic data are predominantly tabular, e.g. credit scoring and medical forecasting (Borisov et al., 2021; Shwartz-Ziv & Armon, 2022). Additionally, nearly 79% of data scientists work with tabular data on a daily basis, compared to only 14% who work with modalities such as images (Kaggle, 2017). We choose a GAN architecture, specifically CTGAN (Xu et al., 2019b), for its widespread use, and its high expected diversity between individually trained models—cf. VAEs, which tend to learn fuzzier distributions (Theis et al., 2015). We use (i) random initialization for each generative model, and (ii) the same real data for training each base model, as this has been shown to perform well in the supervised domain (Lakshminarayanan et al., 2016).

## 4. Empirical Study: the Effect of DGE on Downstream ML

In this section we consider fundamental supervised learning tasks, and how these can be applied in conjunction with synthetic data. We consider using DGE versus the naive synthetic data approach. All experimental details can be found in Appendix A.[1]

### 4.1. Synthetic Data for Model Training

Let us start by considering how we can train better predictive models on synthetic data. We define "better", in terms of a lower generalization error. Choose some predictive loss function $\mathcal{L} : \mathbb{R}^2 \to \mathbb{R}$ and predictor $g_\phi : \mathcal{X} \to \mathcal{Y}$ parameterised by $\phi$. The generalization error is defined as $\mathrm{Err}(g_\phi, p_r) = \mathbb{E}_{p_r}\mathcal{L}(g_\phi(X), Y)$. In the case of classification and a proper scoring rule $\mathcal{L}$, the optimal solution is $g_\phi(x) = p_r(Y|X = x)$.

Because we do not have data from the real distribution, we cannot minimise the error w.r.t. the real distribution directly. Instead, we aim to choose $\phi$ that minimises:

$$\mathbb{E}_\theta[\mathrm{Err}(g_\phi, p_\theta)] = \mathbb{E}_\theta[\mathbb{E}_{(X,Y)\sim p_\theta(X,Y)}L(g_\phi(X), Y))]. \tag{3}$$

The typical synthetic data approach for model training uses a single synthetic dataset. This will yield high variance in the trained model, because we effectively minimise w.r.t. $p_{\theta_1}(Y|X)$ for a single $\theta_1 \sim p(\theta|\mathcal{D}_r)$. Instead, we use an empirical estimate as given by Eq. 2: we train a predictive model on each synthetic dataset individually, and average predictions. Let us show how this improves performance.

**Datasets.** We use a range of datasets with different characteristics of interest: Scikit-learn's Two-moons and Circles toy datasets—simple two-dimensional datasets that we will later use for visualising the learnt models; UCI's Breast Cancer and Adult Census Income (Asuncion & Newman, 2007)—the former a very small dataset such that synthetic data generation is hard, the latter a large dataset with mixed categorical and numerical features; and SEER (Duggan et al., 2016) and Kaggle's Covid-19 dataset (Ministry of Health of Mexico, 2020), two large medical datasets with some highly unbalanced categorical features.

**Set-up.** We first show that downstream models trained on DGE synthetic data, perform better on real data than baselines. We compare against a classifier trained on a single synthetic dataset (Naive (S)) and a pseudo-oracle trained on the real, generative training data ($\mathcal{D}_r$-model). For fair evaluation, we also include the use of an ensemble of classifiers (Naive (E)) and use the same MLP architecture

---

[1]Code and seeded experiments are available at https://github.com/bvanbreugel/deep_generative_ensemble

for all predictive models. At last, we also include a naive generative ensemble approach that concatenates all synthetic datasets (Naive (C)).

We consider the TSTR AUC performance, computed on a hold-out dataset of real data that has not been used for training the generative model. We use CTGAN (Xu et al., 2019b) with the same hyperparameters and architecture in all experiments. In Appendix B we include experiments for other models, showing similar results, and so too do the CelebA and CIFAR-10 results in Appendix D. See Appendix A for experimental details.

**Results.** See Table 2. Training the downstream models on an ensemble of synthetic datasets achieves almost $\mathcal{D}_r$-model performance on real data. In contrast, the naive baseline is often a few percent lower. This performance increase is *not* merely due to ensembles generally being more robust, since the *Naive (ensemble)* method does not perform as well as DGE$_{20}$, despite using 20 base models. Note that the performance of DGE with $K = 20$ is higher on average, but even for $K = 5$ we find a significant advantage over the naive (i.e. $K = 1$) baseline.

These results are unsurprising. When the generative model is erroneous—e.g. it overfits or underfits—we expect the naive method to perform poorer than the DGE method, since the different models in the DGE are unlikely to make the same mistakes. Inevitably the effect of generative model overfitting is dependent on the downstream task. A simpler downstream task—or simpler downstream model—is less prone to copying the generative overfitting. We will explore this further in Section 4.2.2.

Takeaway: By generating multiple synthetic datasets (e.g. $K = 5$), training a prediction model on each, and averaging predictions, one can achieve better performance on real data compared to the naive approach of training a model on a single synthetic dataset. The largest gains are observed when the generative model tends to overfit.

## 4.2. Synthetic Data for Model Evaluation and Selection

Model evaluation and selection (ME/MS) is key to machine learning. ME aims to estimate a model's generalisation error $\text{Err}(g_\phi, p_r)$, while MS aims to choose the model (among a list of models) that minimises this error.

Estimating the generalization error is usually achieved through train-test splits or cross-validation (Hastie et al., 2001). Unfortunately, this approach is not possible when we are in the synthetic data regime *where an ML practitioner has no access to real data*—see Figure 1. The naive approach is to have a single synthetic dataset and treat it like real data—i.e. one trains a predictive model $f$ on $\mathcal{D}_{s,train}^k$ and evaluate it on $\mathcal{D}_{s,test}^k$, for e.g. $k = 1$. This induces bias, since the estimate is taken w.r.t. the same generative model.

It also has a high variance, because it is an estimate w.r.t. a single draw of $\theta_k$. Can we do better?

As a closer alternative to using an independent real test set—which we do not have—we evaluate w.r.t. other test sets, i.e. $\cup_{i \neq k} \mathcal{D}_{s,test}^i$. This reduces the bias and variance, due to us not using the same model parameters $\theta_k$ for training and evaluation. Let us explore in turn how this influences model evaluation and selection.

### 4.2.1. MODEL EVALUATION

**Set-up.** We split the real data into a training and a test set, and as before train $K$ generative models on the training set, providing $\{\mathcal{D}_s^k\}_{k=1}^K$. Subsequently, we split up each $\mathcal{D}_s^k$ into a training and a test set for the downstream model, $\mathcal{D}_{s,train}^k$ and $\mathcal{D}_{s,test}^k$.

We use an MLP as downstream model $g$, trained on a single synthetic training set. We compare the $g$ performance evaluation of the naive approach, our DGE approach and pseudo-oracle evaluation (Oracle)—the latter being the performance of $g$ on a hold-out real dataset. We report results over 20 runs.

**Results.** In Table 3 we see that the DGE and naive evaluation approaches perform very differently. We see the naive approach overestimates the model performance. This is due to a synthetic data variant of data leakage: overfitting in the generative model is copied by the trained model, but is also reflected in the synthetic test set. DGE evaluation suffers less from this bias, since the test set is from a different generative model, i.e. different draws from $\theta^k$. Conversely, generative errors cause DGE



*Figure 3.* Varying generative size for SEER dataset, shows model evaluation becomes overestimated for the naive approach when the generative model starts overfitting. DGE is more robust to this.

to underestimate the downstream performance often. Figure 3 shows the same story by varying the generator complexity. An underfitted generative model affects both approaches similarly—slightly underestimated performances. On the other hand, an overfitted generative model leads to significantly overestimated performance by the naive approach. Let us explore how the downstream task plays its part, through considering different downstream models.

Takeaway: Using train-test-splits on single synthetic datasets can lead to significantly overestimated real-world
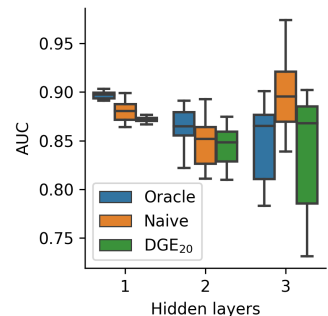
*Table 2.* **Using an ensemble of synthetic datasets for downstream model training improves real-world performance.** AUC performance of different approaches on different datasets, when trained on synthetic data and tested on real data. For the naive methods, we report the median performance across 20 synthetic datasets. Naive (S) uses a single classifier, Naive (E) uses an ensemble of classifiers, though both are trained on a single synthetic dataset. Naive (C) uses all 20 synthetic datasets but naively concatenates them before training a classifier. Note that $DGE_K$ gives consistently better performance on average, even for $K = 5$.

| | Moons | Circles | Adult Income | Breast Cancer | SEER | Covid-19 | Mean |
|---|---|---|---|---|---|---|---|
| $\mathcal{D}_r$-model | $0.996 \pm 0.0$ | $0.868 \pm 0.0$ | $0.87 \pm 0.0$ | $0.993 \pm 0.0$ | $0.907 \pm 0.0$ | $0.928 \pm 0.001$ | 0.927 |
| Naive (S) | $0.981 \pm 0.006$ | $0.801 \pm 0.054$ | $0.821 \pm 0.006$ | $0.975 \pm 0.008$ | $0.885 \pm 0.006$ | $0.869 \pm 0.02$ | 0.889 |
| Naive (E) | $0.981 \pm 0.006$ | $0.802 \pm 0.053$ | $0.837 \pm 0.004$ | $0.978 \pm 0.009$ | $0.888 \pm 0.006$ | $0.895 \pm 0.015$ | 0.897 |
| Naive (C) | $0.985 \pm 0.001$ | $0.862 \pm 0.005$ | $0.852 \pm 0.007$ | $0.974 \pm 0.011$ | $0.906 \pm 0.001$ | $0.895 \pm 0.005$ | 0.912 |
| $DGE_5$ | $0.982 \pm 0.002$ | $0.853 \pm 0.016$ | $0.871 \pm 0.003$ | $0.986 \pm 0.003$ | $0.903 \pm 0.002$ | $0.926 \pm 0.004$ | 0.92 |
| $DGE_{10}$ | $0.983 \pm 0.001$ | $0.861 \pm 0.008$ | $0.883 \pm 0.002$ | $0.986 \pm 0.003$ | $0.906 \pm 0.001$ | $0.935 \pm 0.003$ | 0.926 |
| $DGE_{20}$ | $0.984 \pm 0.001$ | $0.865 \pm 0.003$ | $0.889 \pm 0.001$ | $0.987 \pm 0.003$ | $0.906 \pm 0.001$ | $0.942 \pm 0.001$ | 0.929 |

*Table 3.* **Naive synthetic data model evaluation overestimates real-world performance**. Performance of a fixed model, evaluated using different approaches. The naive approach overestimates performance, whereas the DGE approach slightly underestimates it.

| | Moons | Circles | Adult Income | SEER | Covid-19 | Mean |
|---|---|---|---|---|---|---|
| Oracle | $0.775 \pm 0.14$ | $0.508 \pm 0.036$ | $0.785 \pm 0.015$ | $0.711 \pm 0.108$ | $0.912 \pm 0.014$ | 0.738 |
| Naive | $0.892 \pm 0.072$ | $0.819 \pm 0.132$ | $0.784 \pm 0.028$ | $0.877 \pm 0.061$ | $0.832 \pm 0.042$ | 0.841 |
| $DGE_5$ | $0.703 \pm 0.132$ | $0.518 \pm 0.07$ | $0.773 \pm 0.01$ | $0.743 \pm 0.129$ | $0.819 \pm 0.022$ | 0.711 |
| $DGE_{10}$ | $0.744 \pm 0.139$ | $0.522 \pm 0.094$ | $0.774 \pm 0.01$ | $0.772 \pm 0.088$ | $0.81 \pm 0.017$ | 0.724 |
| $DGE_{20}$ | $0.753 \pm 0.138$ | $0.506 \pm 0.045$ | $0.775 \pm 0.01$ | $0.769 \pm 0.069$ | $0.815 \pm 0.016$ | 0.724 |

performance, due to train and test split both evaluating on the same (potentially erroneous) generative model. On the other hand, DGE tends to underestimate performance, due to uncertainty in the generative model process leading to different—and aggregated fuzzier—label distributions.

### 4.2.2. MODEL SELECTION

**Set-up.** In this part, we ask the question: can we decide on which predictive model to use? We use the same set-up as before, but repeat the experiment for the following models: logistic regression, random forest, 5-nearest neighbour, XGBoost, SVM, and a deep MLP—see Appendix A for experimental details. We consider the ranking of models by different approaches (naive, DGE, and oracle).

**Results.** We see that DGE ranks models similar to the oracle, whereas the naive approach favours complex models. The latter is again explained by the naive approach's positive bias to a model that captures the generative model's overfitting. Congeniality plays a big role this time; the naive approach is inclined to choose a predictive model that is similar to the generative model's parameterisation. Like most deep generative models, CTGAN uses a neural network architecture for implicitly encoding the label distribution, so it is predictable that this can be best replicated by a deep MLP as downstream model.[2] The naive approach becomes

even worse when the amount of synthetic data increases, since for more complex models this allows better learning of the generative label distribution—see Appendix C for experiments.

**Takeaway**: The naive approach consistently selects more complex models that can copy the generative model's $p_\theta(Y|X)$, but which generalise poorly to real data. DGE has a more accurate estimate of real-world performance due to evaluating on other synthetic datasets, which leads to it selecting simpler downstream models that generalise better to real-world data.

### 4.3. Model Uncertainty

Going one step further than evaluation, we look at downstream predictive uncertainty quantification. Using synthetic data like it is real does not account for uncertainty in the generative process itself, which leads to underestimated downstream uncertainty. We focus on classification and define uncertainty in terms of the estimated probability of the predicted label. We show the difference between generative

---

ditional generators like CTGAN usually model the conditional feature distribution $p(X|Y = y)$ using an NN, it is not necessarily true that the output $p(Y|X)$ itself falls in the same model class as the generator. Nonetheless, we do expect the generator output to show similar behaviour (e.g. ReLu artifacts) as the underlying NN, since $p(Y|X) = p(X|Y)p(Y)/(\sum_y p(X|Y = y)p(Y = y))$.

[2]This argument is not entirely straightforward. Since con-

*Table 4.* **Naive evaluation does not preserve real-world model ranking, whereas DGE does.** Model selection on SEER, though evaluating different model classes trained on a downstream synthetic dataset. We see that ranking of models is preserved in all three DGE approaches, in contrast to to the naive approach. Results show mean AUC and standard deviation, taken across 20 runs.

| | deep MLP | SVM | kNN | XGBoost | RF | MLP | Log. reg. | Ranking | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Oracle | $0.767 \pm 0.158$ | $0.781 \pm 0.16$ | $0.8 \pm 0.102$ | $0.838 \pm 0.054$ | $0.844 \pm 0.053$ | $0.849 \pm 0.116$ | $0.869 \pm 0.062$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Naive | $0.849 \pm 0.113$ | $0.804 \pm 0.187$ | $0.823 \pm 0.118$ | $0.885 \pm 0.085$ | $0.873 \pm 0.075$ | $0.867 \pm 0.143$ | $0.866 \pm 0.104$ | 5 | 7 | 6 | 1 | 2 | 3 | 4 |
| $DGE_5$ | $0.797 \pm 0.189$ | $0.819 \pm 0.184$ | $0.831 \pm 0.132$ | $0.859 \pm 0.07$ | $0.87 \pm 0.072$ | $0.873 \pm 0.137$ | $0.884 \pm 0.088$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| $DGE_{10}$ | $0.798 \pm 0.144$ | $0.811 \pm 0.171$ | $0.826 \pm 0.12$ | $0.85 \pm 0.064$ | $0.861 \pm 0.063$ | $0.871 \pm 0.101$ | $0.878 \pm 0.065$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| $DGE_{20}$ | $0.778 \pm 0.139$ | $0.796 \pm 0.159$ | $0.807 \pm 0.11$ | $0.836 \pm 0.064$ | $0.849 \pm 0.057$ | $0.856 \pm 0.104$ | $0.865 \pm 0.071$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 |



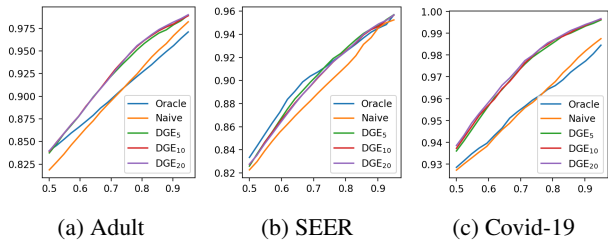(a) Adult      (b) SEER      (c) Covid-19

*Figure 4.* Confidence accuracy curves. Given threshold $\tau$ (x-axis), the y-axis shows the accuracy on test sample with confidence $\hat{P}(Y = \hat{y}|x) > \tau$. DGE achieves consistently higher accuracy for different confidence thresholds.

and predictive uncertainty.

**Set-up.** To separate generative and predictive uncertainty, we include a Deep Ensembles *predictive* model (Lakshminarayanan et al., 2016) that provides uncertainty on the predictive level. Effectively, we compare the sample mean and variance in $\hat{P}(Y = 1|x)$ of (i) a DGE approach, in which each synthetic dataset is used for training a predictive model, (ii) a naive approach, in which one single synthetic dataset is used to train $K$ predictive models in a Deep Ensembles fashion, (iii) Naive (C), in which all synthetic datasets are concatenated and a Deep Ensembles is trained on this dataset. We add toy dataset Gaussian for visualization—see Appendix A for details—and remove the Breast cancer dataset due to insufficient number of real test samples for accurate assessment.

**Results.** First, let us draw the confidence-accuracy curves for different methods on the real-world datasets, see Figure 4. We see that DGE is able to make consistently more confident predictions more successfully than the naive approach. DGE performs en par with the $\mathcal{D}_r$-model, and in some cases outperforms it; this is likely due to the generative models effectively augmenting the data, which has been shown to increase downstream robustness (Antoniou et al., 2017; Dina et al., 2022; Das et al., 2022; Bing et al., 2022).

Let us try to understand why uncertainty quantification on the single synthetic dataset does not suffice, by separating generative and predictive uncertainty. Specifically, let us plot the sample standard deviation of the different classifiers

in the naive Deep Ensembles approach versus the $DGE_{20}$ approach, see Figure 5 and nota bene the different colorbar scales. We include the Naive (C) baseline, which is a naive approach that simply concatenates all synthetic datasets and runs a Deep Ensembles on this, but does not explicitly take into account generative uncertainty. We include this baseline to show that typical generative ensembles that result in a single dataset (see Table 1), fail for UQ. We see that the naive approaches lead to poor diversity between different models within the ensemble. Since they cannot capture the generative uncertainty, these approaches provide little insight into how the model may do on real data. On the other hand, the diversity between $DGE_{20}$ classifiers is much higher and arguably provides more intuitive explanations. We also see that the naive approaches overestimate confidence in low-density regions—even if it is on a decision boundary—whereas DGE does not. This is unsurprising, since generative uncertainty is also highest in these regions.

Takeaway: DGE provides uncertainty on the generative level, which the naive approaches cannot. It is thus essential to ensure individual synthetic datasets in the DGE are published separately (cf. concatenated and shuffled)

### 4.4. Underrepresented Groups

The generative process is expected to be most inaccurate for regions with few samples. Since low-density regions can correspond to minority groups in the population, this would be disconcerting for the reliability of our synthetic data. In this section we explore the quality of downstream models on underrepresented groups in the dataset.

**Set-up.** We investigate the Covid-19 dataset, because it consists of mostly categorical data with unbalanced features. Let us define "underrepresented groups" in terms of minority categories of individual features—see Appendix A. We rerun the experiment from 4.1 and evaluate performance on the minority groups, see Figure 6. We plot the performance relative to a $\mathcal{D}_r$-model, which is trained on $\mathcal{D}_r$ and also evaluated on underrepresented groups.

**Results.** Note the distinctly different behaviour between the naive and DGE approach. The naive approach performs worse than the $\mathcal{D}_r$-model for most underrepresented groups,
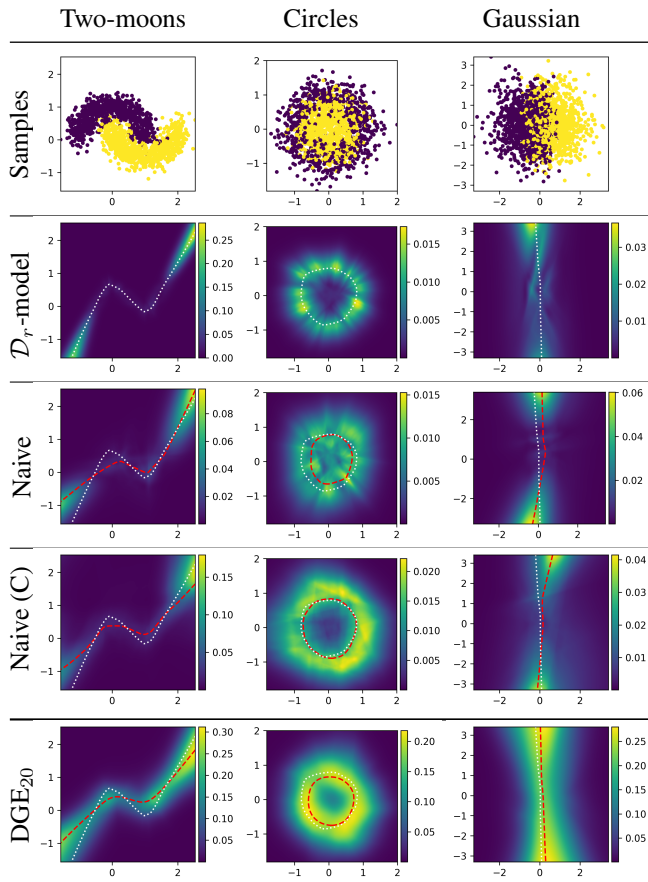
*Figure 5.* Comparison of predictive versus generative uncertainty. We plot the sample std of different ensembles, where columns denote datasets and rows approaches. The $\mathcal{D}_r$ decision boundary ($\hat{P}(Y = 1|x) = 0.5$) is drawn in dotted white and other decision boundaries in dashed red. In almost all cases, these decision boundaries are significantly different. Meanwhile, the std of the naive approaches does not reflect this deviation, hence it underestimates the uncertainty—N.B. the different color scales. This is caused by these methods not considering the generative uncertainty. $\text{DGE}_{20}$ is preferred, as it does reflect generative uncertainty.

even though it performs comparably overall. On the other hand, the DGE approach consistently outperforms the $\mathcal{D}_r$-model. The latter is explained by interpreting DGE as a data augmentation method, in which the synthetic datasets (in this case 20 times the size of the real data) replace the real data. Data augmentation can effectively regularize trained model (Chawla et al., 2002; Zhang et al., 2017; Antoniou et al., 2017), and lead to better performance on underrepresented groups (Bing et al., 2022).

Takeaway: Closer inspection shows that naive downstream model training leads to particularly poor performance on small subgroups in the population. On the other hand, DGE has a regularization effect on the downstream predictor, consistently outperforming the $\mathcal{D}_r$-model on minority groups.
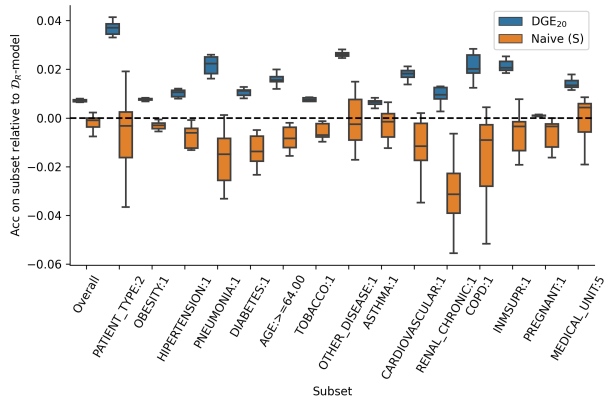


*Figure 6.* Accuracy of downstream model relative to $\mathcal{D}_r$-model, evaluated on minority subsets. The naive approach tends to underperform the $\mathcal{D}_r$-model on minority sets, whereas DGE outperforms the $\mathcal{D}_r$-model.

## 5. Discussion

**DGE.** We have aimed to highlight a gap in our understanding of synthetic data: how to do ML on synthetic data in practice, and the validity of downstream results. We have shown that the standard synthetic data methodology—treating the synthetic data like it is real—leads to poor downstream trained models, evaluation, and uncertainty, with a tendency to do worst for underrepresented groups. We have shown that DGE, which provides multiple synthetic datasets, is a simple and scalable way of avoiding these problems partly. We hope this work will result in a significant change in how synthetic datasets are published, create more interest in synthetic data's use and limitations, and contribute to the trustworthiness of synthetic data. Table 5 highlights the takeaways for practitioners.

**Practical Considerations.** Let us highlight practical considerations. First, DGE is not a perfect approximation for the true posterior of generative model parameters. Further research is required into systemic errors between synthetic datasets, e.g. if the true model is not well approximated by the generator class. Second, the use of ensembles requires extra compute at generation, downstream training, and downstream inference stage. In practice, we have seen that even for $K = 5$ we can get significant gains compared to the naive approach. Additionally, cost may be further reduced by sharing parameters across models or using MC drop-out in the generative model. Third, each synthetic dataset is derived from the same real data, hence there is some data leakage between train and test sets. At last, if privacy is key, the ensemble approach of DGE requires scaling the privacy budget for each synthetic data generator to account for the multiple generators. Please see Appendix E for a longer discussion.

*Table 5.* Guidelines for practitioners.

**Synthetic data publishers**

*(1)* Generate multiple synthetic datasets with different random seeds. This allows better downstream ML, even if the number of generated datasets is small (e.g. $K = 5$). *(2)* These datasets need to be distinguishable—i.e. publishing each set in separate files or create a variable that denotes the set—since for some downstream tasks (e.g. UQ) it does not suffice to simply concatenate datasets without specifying the source of each sample. *(3)* Include meta-data on the generative model, e.g. the amount of training data. Publishing the model class itself is also advisable, since the generative class can have implication on downstream model selection through congeniality.

**Synthetic data users**

*(1)* Do not treat synthetic data like it is real. *(2)* Train models on an ensemble of synthetic datasets, to improve model performance on real data. *(3)* For model evaluation and selection to reflect real-world performance, use train-test splits on the synthetic dataset level: train on some synthetic datasets, and evaluate on others. *(4)* Be careful with UQ in the synthetic data regime, as naive UQ on a single dataset does not consider uncertainty in the generative process itself. By ensembling models trained on different synthetic datasets, one can quantify generative uncertainty. *(5)* All of the above are extra relevant for ML on underrepresented groups, since the generative model is more likely to be inaccurate in these regions.

**Extensions to Other Downstream Tasks.** We have explored using DGE for downstream prediction tasks, but future work could consider other domains (e.g. unsupervised learning, reinforcement learning, statistical analyses).

## Acknowledgments

## References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 12 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2021.05.008.

Alaa, A. M., van Breugel, B., Saveliev, E., and van der Schaar, M. How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models. In *Internal Conference on Machine Learning*, pp. 290–306, 2 2022. URL https://arxiv.org/abs/2102.08921v1.

Antoniou, A., Storkey, A., and Edwards, H. Data Augmentation Generative Adversarial Networks. 11 2017. URL http://arxiv.org/abs/1711.04340.

Asuncion, A. and Newman, D. UCI machine learning repository, 2007.

Bing, S., Dittadi, A., Bauer, S., and Schwab, P. Conditional Generation of Medical Time Series for Extrapolation to Underrepresented Populations. 2022.

Böhm, V., Lanusse, F., and Seljak, U. Uncertainty Quantification with Generative Models. 10 2019. doi: 10.48550/arxiv.1910.10046. URL https://arxiv.org/abs/1910.10046v1.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002.

Das, H. P., Tran, R., Singh, J., Yue, X., Tison, G., Sangiovanni-Vincentelli, A., and Spanos, C. J. Conditional synthetic data generation for robust machine learning applications with limited pandemic data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11792–11800, 2022.

Dina, A. S., Siddique, A., and Manivannan, D. Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *arXiv preprint arXiv:2204.00144*, 2022.

Duggan, M. A., Anderson, W. F., Altekruse, S., Penberthy, L., and Sherman, M. E. The surveillance, epidemiology and end results (SEER) program and pathology: towards strengthening the critical relationship. *The American journal of surgical pathology*, 40(12):e94, 2016.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4004 LNCS:486–503, 2006. ISSN 16113349. doi: 10.1007/11761679{\_}29/COVER. URL https://link.springer.com/chapter/10.1007/11761679_29.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep Ensembles: A Loss Landscape Perspective, 12 2019. URL https://arxiv.org/abs/1912.02757v2.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *33rd International Conference on Machine Learning, ICML 2016*, 3:1651–1660, 6 2015. doi: 10.48550/arxiv.1506.02142. URL https://arxiv.org/abs/1506.02142v6.

Ghosh, A., Kulharia, V., Namboodiri, V., Torr, P. H., and Dokania, P. K. Multi-Agent Diverse Generative Adversarial Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8513–8521, 4 2017. ISSN 10636919. doi: 10.48550/arxiv.1704.02906. URL https://arxiv.org/abs/1704.02906v3.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, 3, 6 2014. doi: 10.1145/3422622.

Grover, A. and Ermon, S. Boosted Generative Models. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 3077–3084, 2 2017. URL https://arxiv.org/abs/1702.08484v2.

Hastie, T., Friedman, J., and Tibshirani, R. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, 2 edition, 2001. ISBN 978-1-4899-0519-2. doi: 10.1007/978-0-387-21606-5. URL http://link.springer.com/10.1007/978-0-387-21606-5.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 2020-December, 6 2020. ISSN 10495258. doi: 10.48550/arxiv.2006.11239. URL https://arxiv.org/abs/2006.11239v2.

Ho, S., Qu, Y., Gu, B., Gao, L., Li, J., and Xiang, Y. DP-GAN: Differentially private consecutive data publishing using generative adversarial nets. *Journal of Network and Computer Applications*, 185:103066, 7 2021. ISSN 1084-8045. doi: 10.1016/J.JNCA.2021.103066.

Hoang, Q., Nguyen, T. D., Le, T., and Phung, D. MGAN: Training Generative Adversarial Nets with Multiple Generators. In *International Conference on Learning Representations*, 2 2018.

Jordon, J., Yoon, J., and Schaar, M. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Jordon, J., Jarrett, D., Saveliev, E., Yoon, J., Elbers, P., Thoral, P., Ercole, A., Zhang, C., Belgrave, D., and van der Schaar, M. Hide-and-Seek Privacy Challenge: Synthetic Data Generation vs. Patient Re-identification. In Escalante, H. J. and Hofmann, K. (eds.), *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pp. 206–215. PMLR, 2 2021. URL https://proceedings.mlr.press/v133/jordon21a.html.

Kaggle. Kaggle machine learning and data science survey, 2017. URL https://www.kaggle.com/datasets/kaggle/kaggle-survey-2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems*, 2017-December:6403–6414, 12 2016. ISSN 10495258. doi: 10.48550/arxiv.1612.01474. URL https://arxiv.org/abs/1612.01474v3.

Liu, D., Jain, M., Dossou, B., Shen, Q., Lahlou, S., Goyal, A., Malkin, N., Emezue, C., Zhang, D., Hassen, N., Ji, X., Kawaguchi, K., and Bengio, Y. GFlowOut: Dropout with Generative Flow Networks. 10 2022. doi: 10.48550/arxiv.2210.12928. URL https://arxiv.org/abs/2210.12928v2.

Ministry of Health of Mexico. COVID-19 Dataset, 4 2020. URL https://www.kaggle.com/datasets/meirnizri/covid19-dataset.

Mordido, G., Yang, H., and Meinel, C. Dropout-GAN: Learning from a Dynamic Ensemble of Discriminators. 7 2018. doi: 10.48550/arxiv.1807.11346. URL https://arxiv.org/abs/1807.11346v2.

Nguyen, T. D., Le, T., Vu, H., and Phung, D. Dual Discriminator Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, 2017-December:2671–2681, 9 2017. ISSN 10495258. doi: 10.48550/arxiv.1709.03831. URL https://arxiv.org/abs/1709.03831v1.

Phan, B., Khan, S., Salay, R., and Czarnecki, K. Bayesian Uncertainty Quantification with Synthetic Data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11699 LNCS:378–390, 2019. ISSN 16113349. doi: 10.1007/978-3-030-26250-1{\_}31/FIGURES/5. URL https://link.springer.com/chapter/10.1007/978-3-030-26250-1_31.

Qian, Z., Cebere, B.-C., and van der Schaar, M. Synthcity: facilitating innovative use cases of synthetic data in different data modalities. 1 2023. doi: 10.48550/arxiv.2301.07573. URL https://arxiv.org/abs/2301.07573v1.

Sensoy, M., Kaplan, L., Cerutti, F., and Saleki, M. Uncertainty-Aware Deep Classifiers Using Generative Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5620–5627, 4 2020. ISSN 2374-3468. doi: 10.1609/AAAI.V34I04.6015. URL https://ojs.aaai.org/index.php/AAAI/article/view/6015.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 5 2022. ISSN 15662535. doi: 10.1016/J.INFFUS.2021.11.011.

Theis, L., Van Den Oord, A., and Bethge, M. A note on the evaluation of generative models. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2015. doi: 10.48550/arxiv.1511.01844. URL https://arxiv.org/abs/1511.01844v3.

Tolstikhin, I., Gelly, S., Bousquet, O., Simon-Gabriel, C. J., and Schölkopf, B. AdaGAN: Boosting Generative Models. *Advances in Neural Information Processing Systems*, 2017-December:5425–5434, 1 2017. ISSN 10495258. URL https://arxiv.org/abs/1701.02386v2.

van Breugel, B. and van der Schaar, M. Beyond privacy: Navigating the opportunities and challenges of synthetic data. *arXiv preprint arXiv:2304.03722*, 2023. URL https://arxiv.org/abs/2305.09235.

van Breugel, B., Kyono, T., Berrevoets, J., and van der Schaar, M. DECAF: Generating Fair Synthetic Data Using Causally-Aware Generative Networks. In *Advances in Neural Information Processing Systems*, 10 2021. doi: 10.48550/arxiv.2110.12884. URL https://arxiv.org/abs/2110.12884v2.

van Breugel, B., Sun, H., Qian, Z., and van der Schaar, M. Membership inference attacks against synthetic data through overfitting detection. In Ruiz, F., Dy, J., and van de Meent, J.-W. (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 3493–3514. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/breugel23a.html.

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. 2 2020. doi: 10.48550/arxiv.2002.06715. URL https://arxiv.org/abs/2002.06715v2.

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter Ensembles for Robustness and Uncertainty Quantification. *Advances in Neural Information Processing Systems*, 2020-December, 6 2020. ISSN 10495258. doi: 10.48550/arxiv.2006.13570. URL https://arxiv.org/abs/2006.13570v3.

Wilson, A. G. and Izmailov, P. Deep Ensembles as Approximate Bayesian Inference, 10 2021. URL https://cims.nyu.edu/~andrewgw/deepensembles/.

Xu, D., Yuan, S., Zhang, L., and Wu, X. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 570–575, 2018.

Xu, D., Wu, Y., Yuan, S., Zhang, L., and Wu, X. Achieving causal fairness through generative adversarial networks. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2019-August, pp. 1452–1458. International Joint Conferences on Artificial Intelligence, 2019a. ISBN 9780999241141. doi: 10.24963/ijcai.2019/201.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling Tabular data using Conditional GAN. *Advances in Neural Information Processing Systems*, 32, 7 2019b. ISSN 10495258. URL https://arxiv.org/abs/1907.00503v2.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019c.

Yoon, J., Jordon, J., and Van Der Schaar, M. Radial-GAN: Leveraging multiple datasets to improve target-specific predictive models using Generative Adversarial Networks. *35th International Conference on Machine Learning, ICML 2018*, 13:9060–9068, 2 2018. URL https://arxiv.org/abs/1802.06403v2.

Yoon, J., Drumright, L. N., and Van Der Schaar, M. Anonymization through data synthesis using generative adversarial networks (ADS-GAN). *IEEE Journal of Biomedical and Health Informatics*, 24(8):2378–2388, 8 2020. ISSN 21682208. doi: 10.1109/JBHI.2020.2980262.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 10 2017. doi: 10.48550/arxiv.1710.09412. URL https://arxiv.org/abs/1710.09412v2.

# A. Experimental Details

## A.1. Data

Throughout the paper we have used Scikit-learn's Two-moons and Circles toy datasets, UCI's Breast Cancer and Adult Census Income (Asuncion & Newman, 2007), and SEER (Duggan et al., 2016) and Mexican Covid-19 dataset (Ministry of Health of Mexico, 2020), two large medical datasets with some highly unbalanced categorical features. For the uncertainty experiment, we have added a third toy dataset *Gaussian*. The dataset was generated using:

$$X_1, X_2 \overset{iid}{\sim} N(0, 1)$$
$$Y \sim Bern(t(X_1 + 1))$$
$$t(x) = \begin{cases} 0 & , x < 0 \\ 1 & , x > 2 \\ \frac{1}{2}x & , otherwise \end{cases}$$

This dataset was chosen because the optimal decision boundary is exact, $\hat{y} = 1(x_1 > 0)$.

## A.2. Experimental Settings

Throughout, we have used CTGAN (Xu et al., 2019c) as the generative model, with hidden layer size 500 for both discriminator and generator. Unless otherwise stated, as downstream model we use Scikit-learn's MLP with 2 hidden layers of size 100, cross-entropy loss. We use learning rate $10^{-3}$, $L_2$ regularization $10^{-4}$, and optimization using Adam for all neural networks.

**Sections 4.1.** For CTGAN we use 2 hidden layers of sizes 500 for all datasets, and we use 2000 examples for training the generative model (except for breast cancer, for which we use 80% of data) and generate 2000 samples.

The pipeline consists of the following.

1. Real data is split into $\mathcal{D}_{r,train}$ and $\mathcal{D}_{r,test}$

2. Generative model $G$ is trained on $\mathcal{D}_{r,train}$ and used to generate. This is repeated $K = 20$ times, for different random seeds.

3. Each generative model is used to generate a respective dataset $\mathcal{D}_s^k$.

4. $\{\mathcal{D}_s^k\}_{k=1}^K$ is used for training the different downstream approaches.

   (a) Naive single (S): the first synthetic dataset is used to train the downstream classifier.

   (b) Naive ensemble (E): the first synthetic dataset is used to train 20 classifiers with different seeds. Outputted probabilities are averaged.

   (c) Naive concatenated (C): synthetic sets $\{\mathcal{D}_s^k\}_{k=1}^2 0$ are concatenated and the full dataset is used for training downstream classifier.

   (d) DGE$_K$: a classifier is trained on each of the synthetic datasets, and the resulting model is the ensemble that averages the outputted probabilities of the first $K$ classifiers.

   (e) $\mathcal{D}_r$-model: this model is trained on the real data, $\mathcal{D}_{r,train}$, and **not** the synthetic data.

5. Each approach's model is tested on $\mathcal{D}_{r,test}$.

6. Steps 2-5 are repeated for 10 runs with different seeds.

**Section 4.2.1.** We increase the generative model and increase the number of training samples. Specifically, we use 5000 samples for training the generative model and generate 5000 samples. We use 3 hidden layers for CTGAN's discriminator and generator, and 2 hidden layers for the downstream model. In the Model Selection experiment, we use default settings for Scikit-learn classifiers—random forest has 100 estimators, kNN uses 5 neighbours, SVM uses an RBF kernel—and use 3 hidden layers for deep MLP and just 1 hiden layer for MLP.

The pipeline is similar as before, with the same steps 1-3.

1. Real data is split into $\mathcal{D}_{r,train}$ and $\mathcal{D}_{r,test}$

2. Generative model $G$ is trained on $\mathcal{D}_{r,train}$ and used to generate. This is repeated $K = 20$ times, for different random seeds.

3. Each generative model is used to generate a respective dataset $\mathcal{D}_s^k$.

4. Each synthetic dataset is split into a train and test split $p_{train} = 0.8$, $\mathcal{D}_{s,train}^k$ and $\mathcal{D}_{s,test}^k$.

5. Each set $\mathcal{D}_s^{train}$ is used for training a downstream model, $g_k$.

6. $\{\mathcal{D}_{s,test}^k\}_{k=1}^K$ are used for evaluating the models $g_k$, using different approaches.

   (a) Naive: $g_k$ is evaluated on $\mathcal{D}_{S,test}^k$.

   (b) DGE$_K$: $g_k$ is evaluated on $\cup_{i \neq k} \mathcal{D}_{s,test}^i$ (limited to $K = 5, 10$ or $20$ other datasets)

   (c) Oracle: this is a pseudo-oracle that evaluates $g_k$ on real test set $\mathcal{D}_r^{test}$.

7. The previous step is repeated for $k = 1, ..., 20$ and results are averaged per approach.

8. (Model selection) For the model selection experiment, steps 4-7 are repeated for the aforementioned model classes (XGBoost, SVM, etc). The target ranking is given by the Oracle model.

**Section 4.3.** We use a similar settings as in Section 4.1—2000 generative training samples, two CTGAN hidden layers and one MLP hidden layer. The MLP uses cross-entropy, which is a proper scoring rule, hence this warrants its use for uncertainty quantification (Lakshminarayanan et al., 2016). The pipeline is as follows.

1. Real data is split into $\mathcal{D}_{r,train}$ and $\mathcal{D}_{r,test}$

2. Generative model $G$ is trained on $\mathcal{D}_{r,train}$ and used to generate. This is repeated $K = 20$ times, for different random seeds.

3. Each generative model is used to generate a respective dataset $\mathcal{D}_s^k$.

4. $\{\mathcal{D}_s^k\}_{k=1}^K$ is used for training the different downstream Deep Ensembles approaches.

    (a) Naive ensemble (E): the first synthetic dataset is used to train 20 classifiers with different seeds.

    (b) Naive concatenated (C): synthetic sets $\{\mathcal{D}_s^k\}_{k=1}^{20}$ are concatenated and the full dataset is used for training 20 downstream classifiers with different seeds.

    (c) DGE$_{20}$: a classifier is trained on each of the synthetic datasets.

    (d) $\mathcal{D}_r$-model: 29 downstream models are trained on the real data, $\mathcal{D}_{r,train}$.

5. We take a grid of test data and create predictions for each approach.

6. For each approach, we take the mean and standard deviation between members of the ensemble.

7. We plot the contour where the mean equals 0.5 (i.e. the decision boundary), and plot the standard deviations over the full grid.

**Section 4.4.** We use the same set-up as Section 4.1—2000 generative training samples with two hidden layers for both the CTGAN and the downstream MLP—however we use a different evaluation set (Step 5 in the pipeline). We take an agnostic approach in defining minorities: we loop through all features and for each feature we choose a minority subset based on this feature alone. For categorical features, we choose the smallest category with less than 20% and more than 0.5% of the population (and skip the feature if no such category exists). For continuous variable age, we choose the 10% oldest patients. Categories are given in the figure behind the feature name.

## B. Type of Generative Model

In the main paper we have used CTGAN for all experiments, but in this section we show main results generalise to other deep generative models. We use the Synthcity library (Qian et al., 2023) and generate data use different architectures, TVAE, a normalizing flow and ADS-GAN (Yoon et al., 2020)—a privacy GAN. We use default settings for each. We repeat experiment 4.1—see Table 6. We see the DGE$_K$ approaches perform significantly better on average, even for $K = 5$.

## C. Effect of Synthetic Dataset Size

In the model selection experiments, we have seen that the naive approach's failure lies not only in the generative process; it is also dependent on the downstream model having the capacity to pick up on generative errors. We have seen that this implies that larger models are more likely to be chosen, since these may learn and copy overfitted correlations in the generated distribution best. In this appendix we explore another actor in the naive approach's failure: the synthetic dataset size.

In Figure C we repeat experiment 4.2.2, but vary the synthetic dataset size from 500 to 20000. The oracle evaluation shows that the downstream trained models become slightly more stable (e.g. deep MLP) when more data is used, but their ranking does not change. The naive approach starts to significantly overestimate the complex models (random forest and deep MLP), giving a very poor and highly dataset-size variable ranking. The DGE approach to model evaluation and selection is much more stable, and follows the oracle closely.

## D. Image Experiment

In the main paper, we focused on tabular data–see Section 3.4 for a motivation—and a GAN-based generator. Evidently, errors in the generative process are not constrained to tabular data. We include preliminary results for CIFAR-10 and CelebA ($64 \times 64$), generated using a small vanilla conditional Denoising Diffusion Probabilistic Model (Ho et al., 2020). For both cases, we follow the same set-up as in Section 4.1, though we use the full training split of the original data for training. For CelebA, we use annotation *male* as label, and generate 10000 samples per class. For CIFAR-10, we generate 6000 samples per class. For the model trained on real data ($\mathcal{D}_r$), we use the full training split. We evaluate on the real test split. See results in Table 7

*Table 6.* Repeating experiment 4.1 for different generative model classes; AUC of downstream model trained on synthetic data, evaluated on real data. $DGE_K$ consistently outperforms the naive approaches on all datasets and generative model classes (except Moons for which it performs similarly).

(a) TVAE

|  | Moons | Circles | Adult Income | Breast Cancer | SEER | COVID-19 | Mean |
|---|---|---|---|---|---|---|---|
| Oracle | 0.996 ± 0.0 | 0.868 ± 0.0 | 0.871 ± 0.0 | 0.993 ± 0.0 | 0.907 ± 0.0 | 0.928 ± 0.001 | 0.927 |
| Naive (S) | 0.989 ± 0.003 | 0.856 ± 0.006 | 0.826 ± 0.012 | 0.97 ± 0.021 | 0.891 ± 0.005 | 0.867 ± 0.019 | 0.9 |
| Naive (E) | 0.989 ± 0.003 | 0.856 ± 0.006 | 0.84 ± 0.009 | 0.979 ± 0.016 | 0.893 ± 0.005 | 0.892 ± 0.013 | 0.908 |
| Naive (C) | 0.991 ± 0.001 | 0.867 ± 0.002 | 0.854 ± 0.004 | 0.975 ± 0.011 | 0.907 ± 0.001 | 0.886 ± 0.005 | 0.913 |
| $DGE_5$ | 0.991 ± 0.001 | 0.866 ± 0.002 | 0.873 ± 0.003 | 0.984 ± 0.006 | 0.906 ± 0.001 | 0.917 ± 0.004 | 0.923 |
| $DGE_{10}$ | 0.991 ± 0.001 | 0.867 ± 0.001 | 0.883 ± 0.002 | 0.986 ± 0.002 | 0.908 ± 0.001 | 0.927 ± 0.002 | 0.927 |
| $DGE_{20}$ | 0.991 ± 0.0 | 0.868 ± 0.001 | 0.89 ± 0.001 | 0.987 ± 0.003 | 0.909 ± 0.0 | 0.934 ± 0.002 | 0.93 |

(b) ADS-GAN

|  | Moons | Circles | Adult Income | Breast Cancer | SEER | COVID-19 | Mean |
|---|---|---|---|---|---|---|---|
| Oracle | 0.996 ± 0.0 | 0.868 ± 0.0 | 0.871 ± 0.001 | 0.993 ± 0.0 | 0.907 ± 0.0 | 0.928 ± 0.001 | 0.927 |
| Naive (S) | 0.981 ± 0.011 | 0.811 ± 0.042 | 0.811 ± 0.012 | 0.961 ± 0.02 | 0.886 ± 0.007 | 0.881 ± 0.021 | 0.888 |
| Naive (E) | 0.981 ± 0.012 | 0.81 ± 0.043 | 0.829 ± 0.009 | 0.972 ± 0.012 | 0.888 ± 0.006 | 0.904 ± 0.012 | 0.897 |
| Naive (C) | 0.989 ± 0.001 | 0.861 ± 0.006 | 0.846 ± 0.003 | 0.965 ± 0.013 | 0.904 ± 0.001 | 0.897 ± 0.007 | 0.91 |
| $DGE_5$ | 0.988 ± 0.003 | 0.852 ± 0.011 | 0.872 ± 0.003 | 0.985 ± 0.006 | 0.902 ± 0.003 | 0.928 ± 0.006 | 0.921 |
| $DGE_{10}$ | 0.988 ± 0.002 | 0.863 ± 0.004 | 0.883 ± 0.002 | 0.986 ± 0.005 | 0.905 ± 0.001 | 0.937 ± 0.003 | 0.927 |
| $DGE_{20}$ | 0.988 ± 0.001 | 0.865 ± 0.002 | 0.889 ± 0.001 | 0.986 ± 0.003 | 0.906 ± 0.001 | 0.943 ± 0.002 | 0.93 |

(c) Normalizing flow

|  | Moons | Circles | Adult Income | Breast Cancer | SEER | COVID-19 | Mean |
|---|---|---|---|---|---|---|---|
| Oracle | 0.996 ± 0.0 | 0.868 ± 0.0 | 0.871 ± 0.0 | 0.993 ± 0.0 | 0.907 ± 0.0 | 0.928 ± 0.001 | 0.927 |
| Naive (S) | 0.984 ± 0.007 | 0.778 ± 0.077 | 0.719 ± 0.041 | 0.966 ± 0.013 | 0.871 ± 0.02 | 0.595 ± 0.116 | 0.819 |
| Naive (E) | 0.984 ± 0.007 | 0.781 ± 0.078 | 0.743 ± 0.038 | 0.971 ± 0.01 | 0.875 ± 0.018 | 0.678 ± 0.123 | 0.839 |
| Naive (C) | 0.986 ± 0.002 | 0.863 ± 0.002 | 0.773 ± 0.02 | 0.934 ± 0.02 | 0.898 ± 0.002 | 0.786 ± 0.039 | 0.873 |
| $DGE_5$ | 0.984 ± 0.005 | 0.857 ± 0.005 | 0.81 ± 0.012 | 0.978 ± 0.009 | 0.899 ± 0.003 | 0.852 ± 0.035 | 0.897 |
| $DGE_{10}$ | 0.984 ± 0.004 | 0.863 ± 0.003 | 0.845 ± 0.008 | 0.983 ± 0.005 | 0.903 ± 0.002 | 0.896 ± 0.019 | 0.912 |
| $DGE_{20}$ | 0.985 ± 0.001 | 0.865 ± 0.001 | 0.865 ± 0.006 | 0.984 ± 0.003 | 0.905 ± 0.001 | 0.923 ± 0.007 | 0.921 |

(a) Oracle AUC      (b) Naive AUC      (c) DGE$_{20}$ AUC

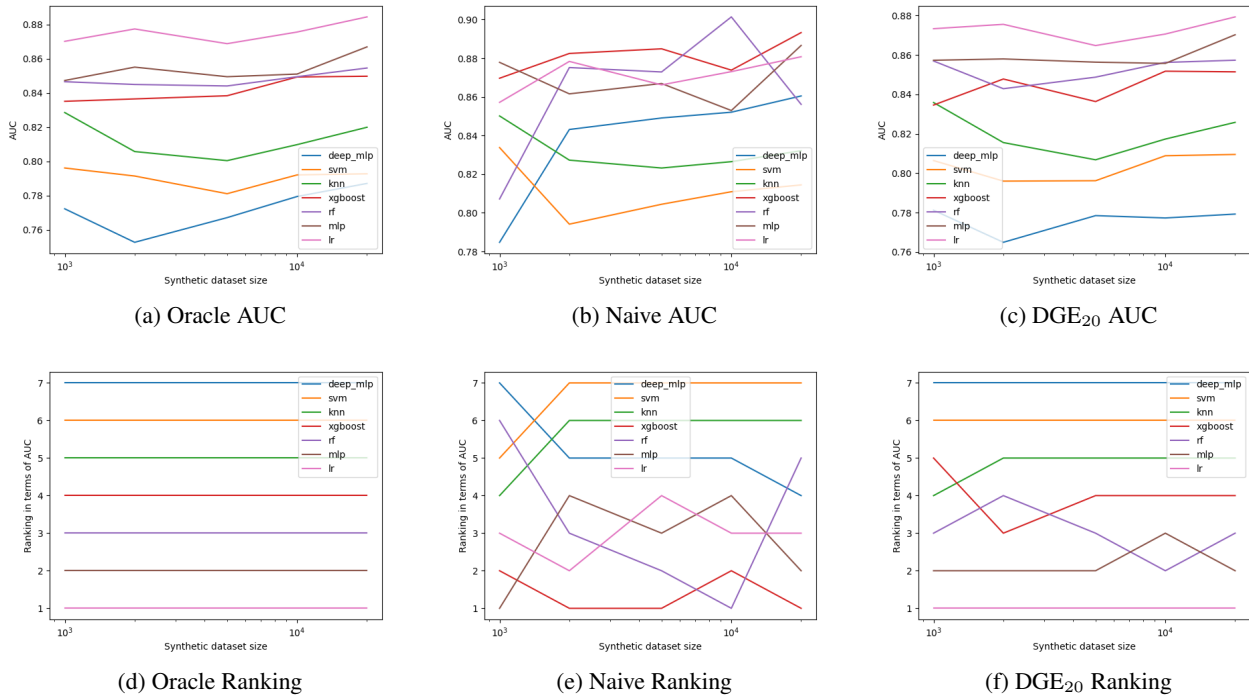(d) Oracle Ranking      (e) Naive Ranking      (f) DGE$_{20}$ Ranking

*Figure 7.* **Varying synthetic dataset size has a large effect on the naive model evaluation and selection**. The naive approach starts overestimating complex models when the dataset size increases, while DGE$_{20}$ follows the oracle much more closely ($\pm 1$ difference). *Ranking: lower is better*

*Table 7.* The advantage of DGE extends to the image domain. Even for small $K$, we can significantly outperform Naive (S) and (C).

| Dataset | CIFAR10 | CelebA |
|---|---|---|
| $\mathcal{D}_r$ | 0.832±0.022 | 0.996±0.000 |
| Naive (S) | 0.662±0.013 | 0.8414±0.116 |
| Naive (C) | 0.690±0.013 | 0.887±0.054 |
| Naive (E) | 0.716±0.004 | **0.927±0.044** |
| DGE$_4$ | **0.725±0.009** | **0.927±0.045** |

## E. Limitations

**Ensembles.** A limitation of DGE is the cost of training and inference scaling linearly with the number of models. In practice, this need not always be a problem. First, we have shown that even for $K = 5$, synthetic datasets in the ensemble vastly outperform the currently-standard naive baseline. Second, generative models can be trained in parallel and so can downstream models. Third, the most expensive part of our experimentation pipeline is the generative stage, which in practice would be performed by a data publisher who runs DGE only once. Fourth, many datasets are not as big as high-resolution image datasets, and are thus cheaper to generate and train models on; training a CTGAN on the Adult

dataset and generating data takes less than four minutes on a consumer PC with an RTX 3080 GPU. Fifth, though we propose a deep generative ensemble with unshared weights across generative models, this can be extended to partly shared weights——e.g. future work could consider a single network with MC-dropout as UQ method (Gal & Ghahramani, 2015).

Besides cost, a deep ensembles are a crude approximation for the true posterior of generative model parameters. The quality of DGE relies partly on how well the base model can approximate the true distribution. In the supervised learning literature, Wenzel et al. (2020) propose an ensemble method based on this idea, that ensembles not just over model parameters, but hyperparameters too. Other methods can be used for approximating the posterior over $\theta$. One can draw inspiration from UQ literature, e.g. use MC dropout UQ (Gal & Ghahramani, 2015), which often performs poorer than DE (Fort et al., 2019), but scales better. Wen et al. (2020) aim to achieve the best of both worlds by sharing parameters across ensemble members, but promoting diversity in model outcome. Future work can extend these methods to the synthetic data generation regime. At last, generating multiple synthetic datasets has privacy implications. If differential privacy guarantees are required, one needs to scale the privacy budget of each synthetic dataset (Dwork et al.,

2006).

**Data Leakage.** In this work we follow the usual Deep Ensembles approach and train each generative model on all real data. Though the original paper (Lakshminarayanan et al., 2016) finds that this performs better for uncertainty estimation, one may wonder whether this does not lead to an underestimate of generalisation error, due to the synthetic test and training sets both being derived from the same real data. This bias can be reduce by training the generators on disjoint subsets of the real data, such that one can train downstream models on some synthetic dataset, and evaluate on another. The downside of this naive approach is that the amount of training data for each generator is just $1/K$ the total amount, and that that by itself could lead to weaker (or more overfitted) generators. In this work we have decided to use all data for training the generators, following (Lakshminarayanan et al., 2016). As seen in Section 4.2.1, there are no signs for a positive bias in model evaluation when using the DGE method, despite all generative models using the same training data. This is possibly explained the hope that generators may overfit to some data, but that different generators overfit in different ways—hence the variance across downstream results from different synthetic datasets is a correct measure of uncertainty. This could motivate increasing independence of generators, e.g. through different model choices and hyperparameters, but we leave this for further work.